This page was intentionally left blank
## Authors

<table>
<thead>
<tr>
<th>Name</th>
<th>Affiliation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thomas Marquart</td>
<td>Uppsala University</td>
</tr>
<tr>
<td>Yves Jung</td>
<td>ESO</td>
</tr>
</tbody>
</table>
This page was intentionally left blank
## Change Record from previous Version

<table>
<thead>
<tr>
<th>Affected Section(s)</th>
<th>Changes/Reason/Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>First Version</td>
</tr>
</tbody>
</table>
This page was intentionally left blank
Contents

1 Introduction to Esoreflex 9

2 Software Installation 10
   2.1 Installing Reflex workflows via macports ........................................ 10
   2.2 Installing Reflex workflows via rpm/yum/dnf ...................................... 10
   2.3 Installing Reflex workflows via install_esoreflex ............................... 11

3 Demo Data 13

4 Quick Start: Reducing The Demo Data 14

5 About the main esoreflex canvas 15
   5.1 Saving And Loading Workflows ....................................................... 15
   5.2 Buttons .......................................................................................... 15
   5.3 Workflow States .............................................................................. 15

6 The CRIRES+ Workflow 16
   6.1 Workflow Canvas Parameters .......................................................... 16
   6.2 Workflow Actors ............................................................................ 17
      6.2.1 Simple Actors ........................................................................... 17
      6.2.2 Lazy Mode ............................................................................... 17

7 Frequently Asked Questions 19
1 Introduction to Esoreflex

This document is a tutorial designed to enable the user to reduce his/her data with the ESO pipeline run under an user-friendly environment, called EsoReflex, concentrating on high-level issues such as data reduction quality and signal-to-noise (S/N) optimisation.

EsoReflex is the ESO Recipe Flexible Execution Workbench, an environment to run ESO VLT pipelines which employs a workflow engine to provide a real-time visual representation of a data reduction cascade, called a workflow, which can be easily understood by most astronomers. The basic philosophy and concepts of Reflex have been discussed by Freudling et al. (2013A&A...559A..96F). Please reference this article if you use Reflex in a scientific publication.

Reflex and the data reduction workflows have been developed by ESO and instrument consortia and they are fully supported. If you have any issue, please have a look to https://support.eso.org to see if this has been reported before or open a ticket for further support.

A workflow accepts science and calibration data, as downloaded from the archive using the CalSelector tool\(^1\) (with associated raw calibrations) and organises them into DataSets, where each DataSet contains one science object observation (possibly consisting of several science files) and all associated raw and static calibrations required for a successful data reduction. The data organisation process is fully automatic, which is a major time-saving feature provided by the software. The DataSets selected by the user for reduction are fed to the workflow which executes the relevant pipeline recipes (or stages) in the correct order. Full control of the various recipe parameters is available within the workflow, and the workflow deals automatically with optional recipe inputs via built-in conditional branches. Additionally, the workflow stores the reduced final data products in a logically organised directory structure employing user-configurable file names.

The workflow follows the "simple way" of reducing calibrations from the CRIRES+ Pipeline User Manual. That is it makes use of the TW-table from the CalibDB and runs a single recipe for each of: darks, flats, wavecal and science reduction.

\(^1\) http://www.eso.org/sci/archive/calselectorInfo.html
2 Software Installation

Esoreflex and the workflows can be installed in different ways: via package repositories, via the install_esoreflex script or manually installing the software tar files.

The recommended way is to use the package repositories if your operating system is supported. The macports repositories support macOS 10.14 to 11, while the rpm/yum repositories support Fedora 28 to 32, CentOS 7, Scientific Linux 7. For any other operating system it is recommended to use the install_esoreflex script.

The installation from package repository requires administrative privileges (typically granted via sudo), as it installs files in system-wide directories under the control of the package manager. If you want a local installation, or you do not have sudo privileges, or if you want to manage different installations on different directories, then use the install_esoreflex script. Note that the script installation requires that your system fulfill several software prerequisites, which might also need sudo privileges.

Reflex 2.11.x needs java JDK 11 to be installed.

Please note that in case of major or minor (affecting the first two digit numbers) Reflex upgrades, the user should erase the $HOME/KeplerData, $HOME/.kepler directories if present, to prevent possible aborts (i.e. a hard crash) of the esoreflex process.

2.1 Installing Reflex workflows via macports

This method is supported for the macOS operating system. It is assumed that macports (http://www.macports.org) is installed. Please read the full documentation at http://www.eso.org/sci/software/pipelines/installation/macports.html.

2.2 Installing Reflex workflows via rpm/yum/dnf

This method is supported for Fedora 28 to 32, CentOS 7, Scientific Linux 7 operating systems, and requires sudo rights. To install, please follow these steps

1. Configure the ESO repository (This step is only necessary if the ESO repository has not already been previously configured).
   - If you are running Fedora, run the following commands:
     ```
sudo dnf install dnf-plugins-core
     ```
   - If you are running CentOS 7, run the following commands:
     ```
sudo yum install yum-utils ca-certificates yum-conf-repos
sudo yum install epel-release
     ```
• If you are running SL 7, run the following commands:

```bash
sudo yum install yum-utils ca-certificates yum-conf-repos
sudo yum install yum-conf-epel
```

2. Install the pipelines

• The list of available top level packages for different instruments is given by:

```bash
sudo dnf list esopipe-\*\-all # (Fedora)
sudo yum list esopipe-\*\-all # (CentOS 7, SL 7)
```

• To install an individual pipeline use the following (This example is for X-Shooter. Adjust the package name to the instrument you require.):

```bash
sudo dnf install esopipe-xshoo-all # (Fedora)
sudo yum install esopipe-xshoo-all # (CentOS 7, SL 7)
```

• To install all pipelines use:

```bash
sudo dnf install esopipe-\*\-all # (Fedora)
sudo yum install esopipe-\*\-all # (CentOS 7, SL 7)
```

For further information, please read the full documentation at http://www.eso.org/sci/software/pipelines/installation/rpm.html.

2.3 Installing Reflex workflows via install_esoreflex

This method is recommended for operating systems other than what indicated above, or if the user has no sudo rights. Software dependencies are not fulfilled by the installation script, therefore the user has to install all the prerequisites before running the installation script.

The software pre-requisites for Reflex 2.11.0 may be found at: http://www.eso.org/sci/software/pipelines/reflex_workflows

To install the Reflex 2.11.0 software and demo data, please follow these instructions:

1. From any directory, download the installation script:

   ```bash
   wget https://ftp.eso.org/pub/dfs/reflex/install_esoreflex
   ```

2. Make the installation script executable:

   ```bash
   chmod u+x install_esoreflex
   ```

3. Execute the installation script:


and the script will ask you to specify three directories: the download directory `<download_dir>`, the software installation directory `<install_dir>`, and the directory to be used to store the demo data `<data_dir>`. If you do not specify these directories, then the installation script will create them in the current directory with default names.

4. Follow all the script instructions; you will be asked whether to use your Internet connection (recommended: yes), the pipelines and demo-datasets to install (note that the installation will remove all previously installed pipelines that are found in the same installation directory).

5. To start Reflex, issue the command:

   `<install_dir>/bin/esoreflex`

It may also be desirable to set up an alias command for starting the Reflex software, using the shell command `alias`. Alternatively, the `PATH` variable can be updated to contain the `<install_dir>/bin` directory.
3 Demo Data

The set of demo data is from a single setting (K2192) of the CRIRES+ spectrograph and includes the following files:

- a flat-field exposure,
- a frame from the U-Ne-lamp,
- a frame from the Fabry-Pérot etalon,
- two exposures of a star, one each from nodding position A and B,
- dark frames that match the DIT for all of the above.
- a TraceWave table that holds the characterization of the echellogram with respect to
  - the location and shape of the spectral orders on the detectors
  - the varying tilt of the slit projection within each order
  - the first guess of the wavelength solution
4 Quick Start: Reducing The Demo Data

Fig. 4.1 shows the layout of the workflow. It is straight-forward in the sense that each recipe-box treats a single kind of raw data, using the results from the earlier steps.
5 About the main esoreflex canvas

5.1 Saving And Loading Workflows

In the course of your data reductions, it is likely that you will customise the workflow for various data sets, even if this simply consists of editing the ROOT_DATA_DIR to a different value for each data set. Whenever you modify a workflow in any way, you have the option of saving the modified version to an XML file using File -> Export As (which will also open a new workflow canvas corresponding to the saved file). The saved workflow may be opened in subsequent esoreflex sessions using File -> Open. Saving the workflow in the default Kepler format (.kar) is only advised if you do not plan to use the workflow with another computer.

5.2 Buttons

At the top of the esoreflex canvas are a set of buttons which have the following functions:

- Zoom in.
- Reset the zoom to 100%.
- Zoom the workflow to fit the current window size (Recommended).
- Zoom out.
- Run (or resume) the workflow.
- Pause the workflow execution.
- Stop the workflow execution.

The remainder of the buttons (not shown here) are not relevant to the workflow execution.

5.3 Workflow States

A workflow may only be in one of three states: executing, paused, or stopped. These states are indicated by the yellow highlighting of the , , and buttons, respectively. A workflow is executed by clicking the button. Subsequently the workflow and any running pipeline recipe may be stopped immediately by clicking the button, or the workflow may be paused by clicking the button which will allow the current actor/recipe to finish execution before the workflow is actually paused. After pausing, the workflow may be resumed by clicking the button again.
6 The CRIRES+ Workflow

The CRIRES+ workflow canvas is organised into a number of areas. From top-left to top-right you will find general workflow instructions, directory parameters, and global parameters. In the middle row you will find five boxes describing the workflow general processing steps in order from left to right, and below this the workflow actors themselves are organised following the workflow general steps.

6.1 Workflow Canvas Parameters

The workflow canvas displays a number of parameters that may be set by the user. Under “Setup Directories” the user is only required to set the \texttt{RAW\_DATA\_DIR} to the working directory for the dataset(s) to be reduced, which, by default, is set to the directory containing the demo data. The \texttt{RAW\_DATA\_DIR} is recursively scanned by the Data Organiser actor for input raw data. The directory \texttt{CALIB\_DATA\_DIR}, which is by default within the pipeline installation directory, is also scanned by the Data Organiser actor to find any static calibrations that may be missing in your dataset(s). If required, the user may edit the directories \texttt{BOOKKEEPING\_DIR}, \texttt{LOGS\_DIR}, \texttt{TMP\_PRODUCTS\_DIR}, and \texttt{END\_PRODUCTS\_DIR}, which correspond to the directories where book-keeping files, logs, temporary products and end products are stored, respectively (see the Reflex User Manual for further details; \cite{ReflexManual}).

There is a mode of the Data Organiser that skips the built-in data organisation and uses instead the data organisation provided by the CalSelector tool. To use this mode, click on \texttt{Use CalSelector associations} in the Data Organiser properties and make sure that the input data directory contains the XML file downloaded with the CalSelector archive request (note that this does not work for all instrument workflows).

Under the “Global Parameters” area of the workflow canvas, the user may set the \texttt{FITS\_VIEWER} parameter to the command used for running his/her favourite application for inspecting FITS files. Currently this is set by default to \texttt{fv}, but other applications, such as \texttt{ds9}, \texttt{skycat} and \texttt{gaia} for example, may be useful for inspecting image data. Note that it is recommended to specify the full path to the visualization application (an alias will not work).

By default the \texttt{EraseDirs} parameter is set to \texttt{false}, which means that no directories are cleaned before executing the workflow, and the recipe actors will work in Lazy Mode (see Section 6.2.2), reusing the previous pipeline recipe outputs if input files and parameters are the same as for the previous execution, which saves considerable processing time. Sometimes it is desirable to set the \texttt{EraseDirs} parameter to \texttt{true}, which forces the workflow to recursively delete the contents of the directories specified by \texttt{BOOKKEEPING\_DIR}, \texttt{LOGS\_DIR}, and \texttt{TMP\_PRODUCTS\_DIR}. This is useful for keeping disk space usage to a minimum and will force the workflow to fully re-reduce the data each time the workflow is run.

The parameter \texttt{RecipeFailureMode} controls the behaviour in case that a recipe fails. If set to \texttt{Continue}, the workflow will trigger the next recipes as usual, but without the output of the failing recipe, which in most of the cases will lead to further failures of other recipes without the user actually being aware of it. This mode might be useful for unattended processing of large number of datasets. If set to \texttt{Ask}, a pop-up window will ask whether the workflow should stop or continue. This is the default. Alternatively, the \texttt{Stop} mode will stop the workflow execution immediately.

The parameter \texttt{ProductExplorerMode} controls whether the ProductExplorer actor will show its window or not. The possible values are \texttt{Enabled}, \texttt{Triggered}, and \texttt{Disabled}. \texttt{Enabled} opens the Produc-
tExplorer GUI at the end of the reduction of each individual dataset. Triggered (default and recommended) opens the ProductExplorer GUI when all the selected datasets have been reduced. Disabled does not display the ProductExplorer GUI.

6.2 Workflow Actors

6.2.1 Simple Actors

Simple actors have workflow symbols that consist of a single (rather than multiple) green-blue rectangle. They may also have an icon within the rectangle to aid in their identification. The following actors are simple actors:

- The **DataOrganiser** actor.
- The **DataSetChooser** actor (inside a composite actor).
- The **FitsRouter** actor Redirects files according to their categories.
- The **ProductRenamer** actor.
- The **ProductExplorer** actor (inside a composite actor).

Access to the parameters for a simple actor is achieved by right-clicking on the actor and selecting Configure Actor. This will open an “Edit parameters” window. Note that the **Product Renamer** actor is a jython script (Java implementation of the Python interpreter) meant to be customised by the user (by double-clicking on it).

There are no composite actors in the current workflow and each box corresponds to a single recipe call.

6.2.2 Lazy Mode

By default, all **RecipeExecuter** actors in a pipeline workflow are “Lazy Mode” enabled. This means that when the workflow attempts to execute such an actor, the actor will check whether the relevant pipeline recipe has already been executed with the same input files and with the same recipe parameters. If this is the case, then the actor will not execute the pipeline recipe, and instead it will simply broadcast the previously generated products to the output port. The purpose of the Lazy Mode is therefore to minimise any reprocessing of data by avoiding data re-reduction where it is not necessary.

One should note that the actor’s Lazy Mode depends on the contents of the directory specified by the parameter **BOOKKEEPING_DIR** and the relevant FITS file checksums. Any modification to the directory contents and/or
the file checksums will cause the corresponding actor to run the pipeline recipe again when executed, thereby re-reducing the input data.

The re-reduction of data at each execution may sometimes be desirable. To force a re-reduction of data for any single RecipeExecuter actor in the workflow, right-click the actor, select Configure Actor, and uncheck the Lazy mode parameter tick-box in the “Edit parameters” window that is displayed. For many workflows the RecipeExecuter actors are actually found inside the composite actors in the top level workflow. To access such embedded RecipeExecuter actors you will first need to open the sub-workflow by right-clicking on the composite actor and then selecting Open Actor.

To force the re-reduction of all data in a workflow (i.e. to disable Lazy mode for the whole workflow), you must uncheck the Lazy mode for every single RecipeExecuter actor in the entire workflow. It is also possible to change the name of the bookkeeping directory, instead of modifying any of the Lazy mode parameters. This will also force a re-reduction of the given dataset(s). A new reduction will start (with the lazy mode still enabled), but the results of previous reduction will not be reused. Alternatively, if there is no need to keep any of the previously reduced data, one can simply set the EraseDirs parameter under the “Global Parameters” area of the workflow canvas to true. This will then remove all previous results that are stored in the bookkeeping, temporary, and log directories before processing the input data, in effect, starting a new clean data reduction and re-processing every input dataset. Note: The option EraseDirs = true does not work in esoreflex version 2.9.x and makes the workflow to crash.
7 Frequently Asked Questions

• The error window fills the whole screen - how can I get to the **Continue**/**Stop** buttons?

Press the Alt key together with your left mouse button to move the window upwards and to the left. At the bottom the **Continue**/**Stop** buttons will be visible. This bug is known but could not yet be fixed.

• I tried to **Open** (or **Configure**) an **Actor** while the workflow is running and now it does not react any more. What should I do?

This is a limitation of the underlying Kepler engine. The only way out is to kill the workflow externally. If you want to change anything while a workflow is running you first need to pause it.

• After a successful reduction of a data set, I changed this data set in some way (e.g. modified or removed some files, or changed the rules of the Data Organizer). When I restart Reflex, the Data Set Chooser correctly displays my new data set, but marks it as “reduced ok”, even though it was never reduced before. What does this mean?

The labels in the column “Reduced” of the Data Set Chooser mark each dataset with “OK”, “Failed” or “-”. These labels indicate whether a data set has previously successfully been reduced at least once, all previous reductions failed, or a reduction has never been tried respectively. Data sets are identified by their name, which is derived from the first science file within the data set. As long as the data set name is preserved (i.e. the first science file in a data set has not changed), the Data Organizer will consider it to be the same data set. The Data Organizer recognizes any previous reductions of data sets it considers to be the same as the current one, and labels the current data set with “OK” if any of them was successful, even if the previously reduced data set differs from the current one.

Note that the Product Explorer will list all the previous reductions of a particular data set only at the end of the reduction. This list might include successful and/or unsuccessful reduction runs with different parameters, or in your case with different input files. The important fact is that these are all reductions of data sets with the same first raw science file. By browsing through all reductions of a particular raw science file, the users can choose the one they want to use.

• Where are my intermediate pipeline products? Intermediate pipeline products are stored in the directory `<TMP_PRODUCTS_DIR>` (defined on the workflow canvas, under Setup Directories) and organised further in directories by pipeline recipe.

• Can I use different sets of bias frames to calibrate my flat frames and science data? Yes. In fact this is what is currently implemented in the workflow(s). Each file in a DataSet has a purpose attached to it ([?]). It is this purpose that is used by the workflow to send the correct set of bias frames to the recipes for flat frame combination and science frame reduction, which may or may not be the same set of bias frames in each case.

• Can I run Reflex from the command line? Yes, use the command:

   esoreflex -n <workflow_path>/<workflow>.xml

   The -n option will set all the different options for Kepler and the workflows to avoid opening any GUI elements (including pipeline interactive windows).
It is possible to specify workflow variables (those that appear in the workflow canvas) in the command line. For instance, the raw data directory can be set with this command:

```
esoreflex -n -RAW_DATA_DIR <raw_data_path> \ 
<workflow_path>/<workflow>.xml
```

You can see all the command line options with the command `esoreflex -h`.

Note that this mode is not fully supported, and the user should be aware that the path to the workflow must be absolute and even if no GUI elements are shown, it still requires a connection to the window manager.

- **How can I add new actors to an existing workflow?** You can drag and drop the actors in the menu on the left of the Reflex canvas. Under Eso-reflex -> Workflow you may find all the actors relevant for pipeline workflows, with the exception of the recipe executer. This actor must be manually instantiated using Tools -> Instantiate Component. Fill in the “Class name” field with org.eso.RecipeExecuter and in the pop-up window choose the required recipe from the pull-down menu. To connect the ports of the actor, click on the source port, holding down the left mouse button, and release the mouse button over the destination port. Please consult the Reflex User Manual ([?]) for more information.

- **How can I broadcast a result to different subsequent actors?** If the output port is a multi-port (filled in white), then you may have several relations from the port. However, if the port is a single port (filled in black), then you may use the black diamond from the toolbar. Make a relation from the output port to the diamond. Then make relations from the input ports to the diamond. Please note that you cannot click to start a relation from the diamond itself. Please consult the Reflex User Manual ([?]) for more information.

- **How can I manually run the recipes executed by Reflex?** If a user wants to re-run a recipe on the command line he/she has to go to the appropriate reflex_book_keeping directory, which is generally reflex_book_keeping/<workflow>/<recipe_name>_<number> There, subdirectories exist with the time stamp of the recipe execution (e.g. 2013-01-25T12:33:53.926/). If the user wants to re-execute the most recent processing he/she should go to the latest directory and then execute the script cmdline.sh. Alternatively, to use a customized esorex command the user can execute

```
ESOREX_CONFIG="INSTALL_DIR/etc/esorex.rc"
PATH_TO/esorex --recipe-config=<recipe>.rc <recipe> data.sof
```

where INSTALL_DIR is the directory where Reflex and the pipelines were installed.

If a user wants to re-execute on the command line a recipe that used a specific raw frame, the way to find the proper data.sof in the bookkeeping directory is via `grep <raw_file> */data.sof`. Afterwards the procedure is the same as before.

If a recipe is re-executed with the command explained above, the products will appear in the directory from which the recipe is called, and not in the reflex_tmp_products or reflex_end_products directory, and they will not be renamed. This does not happen if you use the cmdline.sh script.
• Can I reuse the bookkeeping directory created by previous versions of the pipeline?

In general no. In principle, it could be reused if no major changes were made to the pipeline. However there are situations in which a previously created bookkeeping directory will cause problems due to pipeline versions incompatibility. This is especially true if the parameters of the pipeline recipes have changed. In that case, please remove the bookkeeping directory completely.

• How to insert negative values into a textbox?

Due to a bug in wxPython, the GUI might appear to freeze when attempting to enter a negative number in a parameter's value textbox. This can be worked around by navigating away to a different control in the GUI with a mouse click, and then navigating back to the original textbox. Once focus is back on the original textbox the contents should be selected and it should be possible to replace it with a valid value, by typing it in and pressing the enter key.

• I've updated my Reflex installation and when I run esoreflex the process aborts. How can I fix this problem?

As indicated in Section 2, in case of major or minor (affecting the first two digit numbers) Reflex upgrades, the user should erase the $HOME/KeplerData, $HOME/.kepler directories if present, to prevent possible aborts (i.e. a hard crash) of the esoreflex process.

• How can include my analysis scripts and algorithms into the workflow?

EsoReflex is capable of executing any user-provided script, if properly interfaced. The most convenient way to do it is through the Python actor. Please consult the tutorial on how to insert Python scripts into a workflow available here: www.eso.org/sci/data-processing/Python_and_esoreflex.pdf